```
import numpy as np
import tkinter as tk
from tkinter import messagebox, scrolledtext
from tkinter.ttk import Progressbar

class GaussJordanSolver:
    @staticmethod
    def solve(matrix, constants, verbose=False):
        augmented_matrix = np.hstack((matrix, constants.reshape(-1, 1)))
        steps = []
        rows, cols = augmented_matrix.shape

        for i in range(rows):
            if augmented_matrix[i, i] == 0:
                for j in range(i + 1, rows):
                    if augmented_matrix[j, i] != 0:
                        augmented_matrix[[i, j]] = augmented_matrix[[j, i]]
                        steps.append(f"Swapped rows {i+1} and {j+1}")
                        break
                else:
                    raise ValueError("Matrix is singular or has infinite solutions.")

            augmented_matrix[i] = augmented_matrix[i] / augmented_matrix[i, i]
            steps.append(f"Normalized row {i+1}: {augmented_matrix[i]}")

            for j in range(rows):
                if i != j:
                    factor = augmented_matrix[j, i]
                    augmented_matrix[j] -= factor * augmented_matrix[i]
                    steps.append(f"Eliminated element in row {j+1}, column {i+1}")

        solution = augmented_matrix[:, -1]
        return solution, steps

class LinearEquationApp:
    def __init__(self, master):
        self.master = master
        master.title("Gauss-Jordan Linear Equation Solver")
        master.configure(bg="#34495e")
        self.master.geometry("700x750")

        self.main_frame = tk.Frame(master, bg="#34495e")
        self.main_frame.pack(fill=tk.BOTH, expand=True, padx=20, pady=20)

        # Header Label
        self.header_label = tk.Label(self.main_frame, text="Gauss-Jordan Solver",
font=("Helvetica", 24, "bold"), fg="#ecf0f1", bg="#34495e")
        self.header_label.pack(pady=20)
        # Welcome Message
        self.welcome_label = tk.Label(self.main_frame, text="Solve your system of linear equations using Gauss-Jordan method", font=("Arial", 14), fg="#ecf0f1", bg="#34495e")
        self.welcome_label.pack(pady=5)

        # Input for number of variables
        self.num_vars_label = tk.Label(self.main_frame, text="Enter Number of Variables:", font=("Arial", 12), fg="#ecf0f1", bg="#34495e")
        self.num_vars_label.pack(pady=10)

        self.var_entry = tk.Entry(self.main_frame, width=10, font=("Arial", 14), relief="solid", bd=2)
        self.var_entry.pack(pady=10)

        # Button to create input fields
        self.create_button = tk.Button(self.main_frame, text="Create Inputs", command=self.create_inputs, font=("Arial", 14), bg="#2980b9", fg="white", relief="raised", bd=2)
        self.create_button.pack(pady=15)

        # Input frame to hold coefficient and constant fields
        self.input_frame = tk.Frame(self.main_frame, bg="#ecf0f1")
        self.input_frame.pack(pady=10, fill=tk.X)

        # Solve button
        self.solve_button = tk.Button(self.main_frame, text="Solve", command=self.solve, font=("Arial", 14), state=tk.DISABLED, bg="#27ae60", fg="white", relief="raised", bd=2)
        self.solve_button.pack(pady=10)

        # Progress Bar
        self.progress = Progressbar(self.main_frame, orient=tk.HORIZONTAL, length=300, mode="indeterminate")
        self.progress.pack(pady=20)

        # Output area to show steps and solution
        self.output_area = scrolledtext.ScrolledText(self.main_frame, width=75, height=15, font=("Courier New", 12), wrap=tk.WORD, bd=3, relief="sunken")
        self.output_area.pack(pady=10)

        # Footer
        self.footer_label = tk.Label(self.main_frame, text="Gauss-Jordan Solver © 2025", font=("Arial", 10), fg="#ecf0f1", bg="#34495e")
        self.footer_label.pack(side=tk.BOTTOM, pady=15)

    def create_inputs(self):
        try:
            num_vars = int(self.var_entry.get())
            if num_vars <= 0:
                raise ValueError
        except ValueError:
            messagebox.showerror("Invalid Input", "Please enter a positive integer.")
            return

        for widget in self.input_frame.winfo_children():
            widget.destroy()

        self.entries = []
        tk.Label(self.input_frame, text="Coefficients:", font=("Arial", 12), bg="#ecf0f1").grid(row=0, column=0, columnspan=num_vars, padx=10)

        for i in range(num_vars):
            row_entries = []
            for j in range(num_vars + 1):
                entry = tk.Entry(self.input_frame, width=8, font=("Arial", 12), relief="solid", bd=2)
                entry.grid(row=i + 1, column=j, padx=5, pady=5)
                row_entries.append(entry)
            self.entries.append(row_entries)

        # Enable solve button after inputs are created
        self.solve_button.config(state=tk.NORMAL)

    def solve(self):
        # Start progress bar animation
        self.progress.start()

        try:
            matrix = []
            constants = []
            for row in self.entries:
                matrix.append([float(entry.get()) for entry in row[:-1]])
                constants.append(float(row[-1].get()))
            matrix = np.array(matrix, dtype=float)
            constants = np.array(constants, dtype=float)
        except ValueError:
            messagebox.showerror("Invalid Input", "Please fill in all fields with valid numbers.")
            self.progress.stop()
            return

        try:
            solution, steps = GaussJordanSolver.solve(matrix, constants, verbose=True)
            self.output_area.delete(1.0, tk.END)
            self.output_area.insert(tk.END, "Steps to solve:\n")
            for step in steps:
                self.output_area.insert(tk.END, step + "\n")
            self.output_area.insert(tk.END, "\nSolution:\n")
            for i, value in enumerate(solution):
                self.output_area.insert(tk.END, f"x{i+1} = {value:.4f}\n")
        except ValueError as e:
            messagebox.showerror("Error", str(e))

        # Stop progress bar animation after solution is found
        self.progress.stop()

if __name__ == "__main__":
    root = tk.Tk()
    app = LinearEquationApp(root)
    root.mainloop()
```

https://youtu.be/0kcNC-l3HVw?si=l8JWjFiSBPz78NEd